ECE684: NLP Final Project

Harsh Bandhey

In [166…

```
import nltk
import random
import pickle
import numpy as np
import pandas as pd
from spacy import displacy
from tqdm.notebook import tqdm
from tqdm.keras import TqdmCallback
from sklearn.metrics import classification_report
from keras.utils import pad_sequences
from keras.models import Sequential
from keras.utils.vis_utils import plot_model
from keras.layers import Dense, Dropout, TimeDistributed
from keras.layers import Embedding, LSTM, Bidirectional, CuDNNLSTM
from keras.callbacks import TensorBoard
from nltk.probability import LidstoneProbDist
from nltk.probability import SimpleGoodTuringProbDist, WittenBellProbDist
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_theme()
```

Helper Functions

In [167…

```
#@title Data Loading Functions
def conll_sentences(conll_file):
    sent = []
    pos = []
    chunk = []
    entity = []
    temp_sent = []
    temp_pos = []
    temp_chunk = []
    temp_entity = []

    with open(conll_file) as f:
        conll_raw_data = f.readlines()
    conll_raw_data = [x.strip() for x in conll_raw_data]

    for line in conll_raw_data:
        if line != '':
            split_line = line.split()
            if len(split_line) == 4:
                if split_line[0] != '-DOCSTART-':
                    temp_sent.append(split_line[0])
                    temp_pos.append(split_line[1])
                    temp_chunk.append(split_line[2])

                    # Rename entity values as PER, LOC, ORG, MISC, O
                    old_ent = split_line[3]
                    if old_ent in ('I-ORG', 'B-ORG'):
                        new_ent = 'ORG'
                    elif old_ent in ('I-LOC', 'B-LOC'):
                        new_ent = 'LOC'
                    elif old_ent in ('I-MISC', 'B-MISC'):
                        new_ent = 'MISC'
                    elif old_ent in ('I-PER', 'B-PER'):
                        new_ent = 'PER'
                    else:
                        new_ent = 'O'
                    temp_entity.append(new_ent)
            else:
                raise IndexError('Line split length does not equal 4.')
        else:
            if len(temp_sent) > 0:
                assert(len(sent) == len(pos))
                assert(len(sent) == len(chunk))
                assert(len(sent) == len(entity))
                sent.append(temp_sent)
                pos.append(temp_pos)
                chunk.append(temp_chunk)
                entity.append(temp_entity)
                temp_sent = []
                temp_pos = []
                temp_chunk = []
                temp_entity = []

    return sent, pos, chunk, entity
```

In [168…

```
#@title Evaluation Function
def get_preds(model, test_data):
    preds_list, actual_list = list(), list()
    for X in tqdm(test_data):
        sent = [i[0] for i in X]
        tagged = model.tag(sent)
        preds = [i[1] for i in tagged]
        act = [i[1] for i in X]
        preds_list += preds
        actual_list += act
    return preds_list, actual_list

def accuracy(expected, predicted):
    total = 0
    correct = 0
    for i in range(len(expected)):
        total += 1
        if (expected[i] == predicted[i]):
            correct += 1
    acc = correct/total
    print('accuracy = {} / {} = {}'.format(correct, total, round(acc*100,2)))
    return acc

def dispacy_doc(item, title):
    cur = 0
    ents = []
    for a, b in item:
        ents.append({"start":cur,
             "end":cur+len(a),
                "label": b })
            cur = cur+len(a)+1
    element = { "text": " ".join([a for a,b in item]),
                "ents":  ents,
                "title": title }
    # displacy.render(doc, style="ent", manual=True, jupyter=True)
    return element
```

In [169…

```
#@title Encoding
def encode(entities_list):
    encoded_list = []
    # one-hot formatting: [PER LOC ORG MISC O]
    for entities in entities_list:
        encoded_vectors = []
        for ent in entities:
            if ent == 'PER':
                encoded_vectors.append([1, 0, 0, 0, 0])
            elif ent == 'LOC':
                encoded_vectors.append([0, 1, 0, 0, 0])
            elif ent == 'ORG':
                encoded_vectors.append([0, 0, 1, 0, 0])
            elif ent == 'MISC':
                encoded_vectors.append([0, 0, 0, 1, 0])
            else:
                encoded_vectors.append([0, 0, 0, 0, 1])
        encoded_list.append(encoded_vectors)
    return encoded_list

def decode(scores_list):
    predictions_list = []
    for predictions in scores_list:
        decoded = []
        for pred in predictions:
            max_index = np.argmax(pred)
            if max_index == 0:
                decoded.append('PER')
            elif max_index == 1:
                decoded.append('LOC')
            elif max_index == 2:
                decoded.append('ORG')
            elif max_index == 3:
                decoded.append('MISC')
            elif max_index == 4:
                decoded.append('O')
        predictions_list.append(decoded)
    return predictions_list
```

In [170…

```
#@title Glove Emedding Functions
def load_glove_dict(glove_file):
    word_dict = {}
    with open(glove_file, 'r') as f:
        for line in f:
            split = line.split()
            word = split[0]
            vector = np.array([float(v) for v in split[1:]])
            word_dict[word] = vector
    return word_dict
```

```
def get_glove_vector(g_dict, word):
    try:
        vector = g_dict[word.lower()]
    except KeyError:
        vector_len = len(g_dict['test'])
        vector = np.array([0.]*vector_len)
    return vector
```

In [171…

```
#@title Plot Functions
def plot_acc(history):
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title('Bi-LSTM Accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epochs')
    plt.legend(['train', 'Test'], loc='upper left')
    plt.show()

def plot_loss(history):
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Bi-LSTM Loss')
    plt.ylabel('Loss')
    plt.xlabel('Epochs')
    plt.legend(['Train', 'Test'], loc='upper right')
    plt.show()
```

Data Processing

In [7]:

```
!mkdir dataset
!wget -q "https://drive.google.com/uc?export=download&id=1FQ8ZBCYxxrhGOhnnyi3Qzo
!wget -q "https://drive.google.com/uc?export=download&id=1lf3hKR4ndPbJcw3QmMI-Uw
!wget -q "https://drive.google.com/uc?export=download&id=1EhIJKhIA4DaBTcTJpszPGZ
!wget -q "https://drive.google.com/uc?export=download&id=1mOBEvOino44PZ8UR5dOlR1
```

In [6]:

```
train_file = './dataset/eng.train'
testa_file = './dataset/eng.testa'
testb_file = './dataset/eng.testb'
testc_file = './dataset/eng.testc'
```

In [7]:

```
entity_set = {'PER', 'LOC', 'ORG', 'MISC', 'O'}
train_sent, train_pos, train_chunks, train_entities = conll_sentences(train_file
testa_sent, testa_pos, testa_chunks, testa_entities = conll_sentences(testa_file
testb_sent, testb_pos, testa_chunks, testb_entities = conll_sentences(testb_file
testc_sent, testc_pos, testa_chunks, testc_entities = conll_sentences(testc_file
```

In [8]:

```
combined_sentences = train_sent + testa_sent + testb_sent + testc_sent
word_set = set()
for sent in combined_sentences:
    for word in sent:
        word_set.add(word)
```

In [9]:
```python
train_data = list(zip(train_sent, train_entities))
train_data = [list(zip(x,y)) for x,y in train_data]
```

In [10]:
```python
test_sent, test_entities = testa_sent+testb_sent, testa_entities+testb_entities
test_data = list(zip(test_sent, test_entities))
test_data = [list(zip(x,y)) for x,y in test_data]
```

HMM Model

Training and Evaluation

In [67]:
```python
trainer = nltk.tag.HiddenMarkovModelTrainer(states=entity_set, symbols=word_set)
estimator = lambda fd, bins: LidstoneProbDist(fd, 0.1, bins)
model = trainer.train_supervised(train_data, estimator=estimator)
```

In [109…
```python
y_true, y_pred = get_preds(model, test_data)
print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support

         LOC       0.80      0.88      0.84      3665
        MISC       0.75      0.86      0.80      1903
           O       0.99      0.95      0.97     84102
         ORG       0.67      0.85      0.74      3611
         PER       0.68      0.89      0.77      4516

    accuracy                           0.94     97797
   macro avg       0.78      0.89      0.83     97797
weighted avg       0.95      0.94      0.95     97797
```

Running a few examples on the HMM Model

In [78]:
```python
item = test_data[3]
doc = dispacy_doc(item, "Orignal NER")
html = displacy.render(doc, style="ent", manual=True, jupyter=True)
```

Orignal NER

Their O stay O on O top O , O though O , O may O be O short-lived O as O title O rivals O Essex ORG , O Derbyshire ORG and O Surrey ORG all O closed O in O on O victory O while O Kent ORG made O up O for O lost O time O in O their O rain-affected O match O against O Nottinghamshire ORG . O

In [79]:
```
_y_pred = model.tag([a for a,b in item])
doc = dispacy_doc(_y_pred, "Predicted NER")
html = displacy.render(doc, style="ent", manual=True, jupyter=True)
```

Predicted NER

Their O stay O on O top O , O though O , O may O be O short-lived O as O title O rivals O Essex ORG , O Derbyshire ORG and ORG Surrey ORG all O closed O in O on O victory O while O Kent ORG made O up O for O lost O time O in O their O rain-affected O match O against O Nottinghamshire ORG . O

Data Generation From HMM

In [68]:
```
synth_train_data  = [model.random_sample(random.Random(),
                     np.random.randint(10, 25)) for i in tqdm(range(len(train_dat
synth_test_data  = [model.random_sample(random.Random(),
                     np.random.randint(10, 25)) for i in tqdm(range(len(test_data
```

In [76]:
```
with open('syndata.pkl', 'wb')  as f:
    obj_dict = {"synth_train_data":synth_train_data,
                "synth_test_data":synth_test_data}
    pickle.dump(obj_dict, f)
```

In [70]:
```
with open('syndata.pkl', 'rb')  as f:
    obj_dict = pickle.load(f)
synth_train_data = obj_dict["synth_train_data"]
synth_test_data = obj_dict["synth_test_data"]
```

In [28]:
```
item = synth_train_data[2]
doc = dispacy_doc(item, "Generated NER Data")
html = displacy.render(doc, style="ent", manual=True, jupyter=True)
```

Generated NER Data

Commonwealth ORG 17 O with O either O billions O . O were O military O who O The O companies O

Testing HMM on Synthetic Data

In [53]:
```
trainer = nltk.tag.HiddenMarkovModelTrainer(states=entity_set, symbols=word_set)
estimator = lambda fd, bins: SimpleGoodTuringProbDist(fd, bins=1e5)
model = trainer.train_supervised(synth_train_data, estimator=estimator)
```

```
/usr/local/lib/python3.8/dist-packages/nltk/probability.py:1452: UserWarning: Si
mpleGoodTuring did not find a proper best fit line for smoothing probabilities o
f occurrences. The probability estimates are likely to be unreliable.
  warnings.warn(
```

In [102…

```
y_true, y_pred = get_preds(model, synth_test_data)
print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support

         LOC       0.46      0.75      0.57       265
        MISC       0.39      0.57      0.46       197
           O       0.97      0.93      0.95      9866
         ORG       0.55      0.64      0.59       466
         PER       0.58      0.72      0.64       497

    accuracy                           0.90     11291
   macro avg       0.59      0.72      0.64     11291
weighted avg       0.92      0.90      0.90     11291
```

In [165…

```
idx = 3
item = synth_test_data[idx]
doc = dispacy_doc(item, "Orignal NER")
html = displacy.render(doc, style="ent", manual=True, jupyter=True)
```

Orignal NER

finally O the O Two O of O , O (O Friday O AND O is O who O the O Jerry PER Bevan PER David PER 0.38 O Hungary LOC of O the O surprise O

In [164…

```
item = synth_test_data[idx]
_y_pred = model.tag([a for a,b in item])
doc = dispacy_doc(_y_pred, "Predicted NER")
html = displacy.render(doc, style="ent", manual=True, jupyter=True)
```

Predicted NER

finally O the O Two O of O , O (O Friday O AND O is O who O the O Jerry PER Bevan PER David PER 0.38 O Hungary LOC of O the O surprise O

Netural Network (BiLSTM) Model

Data Prep for LSTM

In [172…

```
max_sent_len = len(max(combined_sentences, key=len))
word_dict = dict()
for i, wrd in enumerate(word_set):
    word_dict[wrd] = int(i) + 1
```

In [173…

```
max_features = len(word_set) + 1
maxlen = max_sent_len
batch_size = 256
output_dim = 100
epochs = 30
drop_out = 0.05
n_tags = 5
```

In [174…

```
def word2vec(sentences):
    out = []
    for sent in sentences:
        temp = []
        for word in sent:
            temp.append(word_dict[word])
        out.append(temp)
    return np.array(out)
```

In [175…

```
test_sent = testa_sent+testb_sent
train_sent_vec = pad_sequences(word2vec(train_sent), maxlen=maxlen, padding='pos
test_sent_vec = pad_sequences(word2vec(test_sent), maxlen=maxlen, padding='post'
test_entities = testa_entities+testb_entities
train_ent_vec = pad_sequences(encode(train_entities), maxlen=maxlen, padding='po
test_ent_vec = pad_sequences(encode(test_entities), maxlen=maxlen, padding='post
```

```
<ipython-input-174-74d0e4c39a73>:8: VisibleDeprecationWarning: Creating an ndarr
ay from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or
ndarrays with different lengths or shapes) is deprecated. If you meant to do thi
s, you must specify 'dtype=object' when creating the ndarray.
  return np.array(out)
```

LSTM Models

In [20]:

```
!wget https://worksheets.codalab.org/rest/bundles/0x4090ba96b8a444c2a44b2c47884c
```

```
--2022-12-18 13:33:44--  https://worksheets.codalab.org/rest/bundles/0x4090ba96b
8a444c2a44b2c47884c25f2/contents/blob/glove.twitter.27B.100d.txt
Resolving worksheets.codalab.org (worksheets.codalab.org)... 13.68.212.115
Connecting to worksheets.codalab.org (worksheets.codalab.org)|13.68.212.115|:44
3... connected.
HTTP request sent, awaiting response... 200 OK
Syntax error in Set-Cookie: codalab_session=""; expires=Thu, 01 Jan 1970 00:00:0
0 GMT; Max-Age=-1; Path=/ at position 70.
Length: unspecified [text/plain]
Saving to: ‘glove.twitter.27B.100d.txt.1’

glove.twitter.27B.1     [              <=>    ] 974.34M   104MB/s    in 9.7s

2022-12-18 13:33:54 (100 MB/s) - ‘glove.twitter.27B.100d.txt.1’ saved [102166937
9]
```

In [33]:
```
def get_glove_weights():
    glove_file='glove.twitter.27B.100d.txt'
    g_dict = load_glove_dict(glove_file)
    embedding_matrix = np.zeros((max_features, output_dim))
    for word, i in tqdm(word_dict.items()):
        embedding_vector = g_dict.get(word)
        if embedding_vector is not None:
            # words not found in embedding index will be all-zeros.
            embedding_matrix[i] = embedding_vector
    return embedding_matrix
```

In [34]:
```
embedding_matrix = get_glove_weights()
```

In [35]:
```
def get_bilstm_lstm_model():
    model = Sequential()
    model.add(Embedding(max_features, output_dim, input_length=maxlen,
                        weights=[embedding_matrix], mask_zero=True,
                        trainable=False))
    model.add(Bidirectional(LSTM(units=output_dim,
                                 return_sequences=True, dropout=drop_out,
                                 recurrent_dropout=drop_out),
                                 merge_mode = 'concat'))
    model.add(LSTM(units=output_dim, return_sequences=True, dropout=drop_out*2,
                   recurrent_dropout=drop_out*2))
    model.add(TimeDistributed(Dense(n_tags, activation="softmax")))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['a
    # model.summary()
    return model
```

Model Training

In [36]:
```
GLOVE = True
batch_size = 512
```

In [37]:
```
if GLOVE:
    model = get_bilstm_lstm_model()
    train_vec = train_sent_vec
    test_vec = test_sent_vec
else:
    pass
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet t
he criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet t
he criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet t
he criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.
```

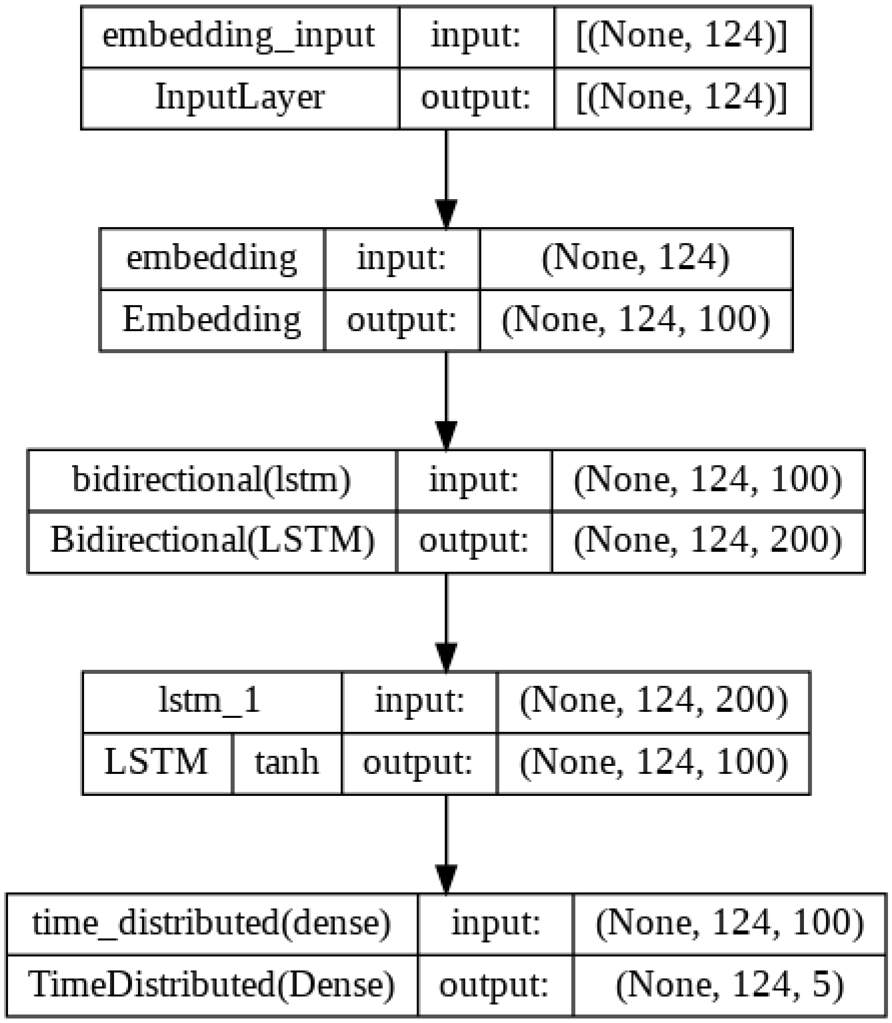
In [38]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 124, 100)          3029000   
                                                                 
 bidirectional (Bidirectiona  (None, 124, 200)         160800    
 l)                                                              
                                                                 
 lstm_1 (LSTM)               (None, 124, 100)          120400    
                                                                 
 time_distributed (TimeDistr  (None, 124, 5)           505       
 ibuted)                                                         
                                                                 
=================================================================
Total params: 3,310,705
Trainable params: 281,705
Non-trainable params: 3,029,000
_________________________________________________________________
```

In [181…
```
plot_model(model,
           show_shapes=True,
           show_layer_activations=True)
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet t
he criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet t
he criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet t
he criteria. It will use a generic GPU kernel as fallback when running on GPU.
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet
the criteria. It will use a generic GPU kernel as fallback when running on GPU.
```

Out[181…

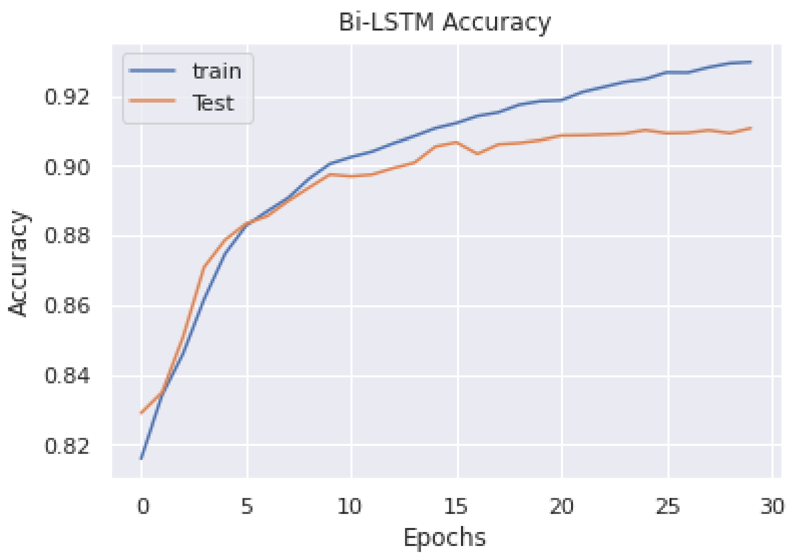


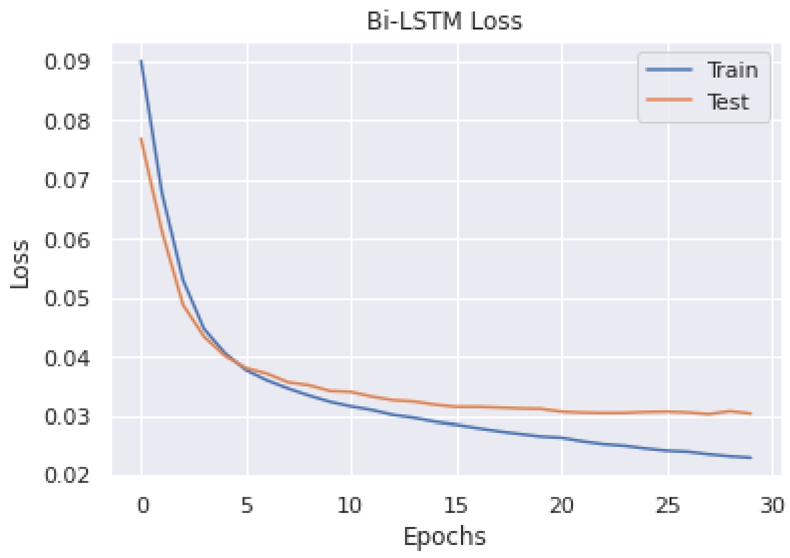
In [40]:

```
history = model.fit(train_vec, train_ent_vec,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=[test_vec, test_ent_vec],
          verbose=0, callbacks=[TqdmCallback(verbose=1)])
```

In [43]:

```
plot_acc(history)
plot_loss(history)
```





In [44]:

```
y_true, y_pred = [], []
test_pred = decode(model.predict(test_vec))
for i in range(len(test_entities)):
    y_true+=test_entities[i]
    y_pred+=test_pred[i][:len(test_entities[i])]
print(classification_report(y_true, y_pred))
```

```
210/210 [==============================] - 24s 110ms/step
              precision    recall  f1-score   support

         LOC       0.67      0.46      0.55      4019
        MISC       0.51      0.43      0.47      2186
           O       0.95      0.97      0.96     81082
         ORG       0.61      0.54      0.58      4588
         PER       0.76      0.82      0.79      5922

    accuracy                           0.91     97797
   macro avg       0.70      0.65      0.67     97797
weighted avg       0.90      0.91      0.91     97797
```

In [45]:
```
idx = 3
_x_wrd, _x_lab, _x_vec = test_sent[idx], test_entities[idx], test_vec[idx]
_y_pred = decode(model.predict(np.array([_x_vec,])))[0][:len(_x_wrd)]
org = [(_x_wrd[i], _x_lab[i]) for i in range(len(_x_lab))]
item = [(_x_wrd[i], _y_pred[i]) for i in range(len(_x_wrd))]
```

```
1/1 [==============================] - 0s 198ms/step
```

In [46]:
```
doc = dispacy_doc(org, "Orignal NER")
html = displacy.render(doc, style="ent", manual=True, jupyter=True)
```

Orignal NER

Their O stay O on O top O , O though O , O may O be O
short-lived O as O title O rivals O Essex ORG , O Derbyshire ORG
and O Surrey ORG all O closed O in O on O victory O while O
Kent ORG made O up O for O lost O time O in O their O
rain-affected O match O against O Nottinghamshire ORG . O

In [47]:
```
doc = dispacy_doc(item, "Generated NER")
html = displacy.render(doc, style="ent", manual=True, jupyter=True)
```

Generated NER

Their O stay O on O top O , O though O , O may O be O
short-lived O as O title O rivals O Essex ORG , O Derbyshire ORG
and O Surrey ORG all O closed O in O on O victory O while O
Kent PER made O up O for O lost O time O in O their O
rain-affected O match O against O Nottinghamshire LOC . O

LSTM Model on Synthetic Dataset

In [99]:
```
maxlen=25
synth_train_sent = [[x[0] for x in sent] for sent in synth_train_data]
synth_test_sent = [[x[0] for x in sent] for sent in synth_test_data]
train_sent_vec = pad_sequences(word2vec(synth_train_sent), maxlen=maxlen, paddin
test_sent_vec = pad_sequences(word2vec(synth_test_sent), maxlen=maxlen, padding=
train_entities = [[x[1] for x in sent] for sent in synth_train_data]
test_entities = [[x[1] for x in sent] for sent in synth_test_data]
train_ent_vec = pad_sequences(encode(train_entities), maxlen=maxlen, padding='po
                              value= [0, 0, 0, 0, 1])
test_ent_vec = pad_sequences(encode(test_entities), maxlen=maxlen, padding='post
                              value= [0, 0, 0, 0, 1])
```

```
<ipython-input-31-74d0e4c39a73>:8: VisibleDeprecationWarning: Creating an ndarra
y from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or n
darrays with different lengths or shapes) is deprecated. If you meant to do thi
s, you must specify 'dtype=object' when creating the ndarray.
  return np.array(out)
```

In [125…

```
if GLOVE:
    model = get_bilstm_lstm_model()
    train_vec = train_sent_vec
    test_vec = test_sent_vec
else:
    pass
```

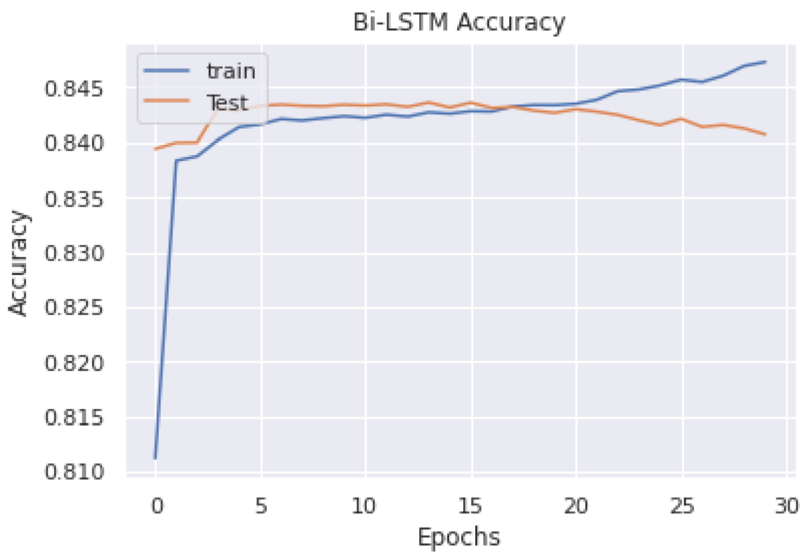
```
WARNING:tensorflow:Layer lstm_30 will not use cuDNN kernels since it doesn't mee
t the criteria. It will use a generic GPU kernel as fallback when running on GP
U.
WARNING:tensorflow:Layer lstm_30 will not use cuDNN kernels since it doesn't mee
t the criteria. It will use a generic GPU kernel as fallback when running on GP
U.
WARNING:tensorflow:Layer lstm_30 will not use cuDNN kernels since it doesn't mee
t the criteria. It will use a generic GPU kernel as fallback when running on GP
U.
WARNING:tensorflow:Layer lstm_31 will not use cuDNN kernels since it doesn't mee
t the criteria. It will use a generic GPU kernel as fallback when running on GP
U.
```

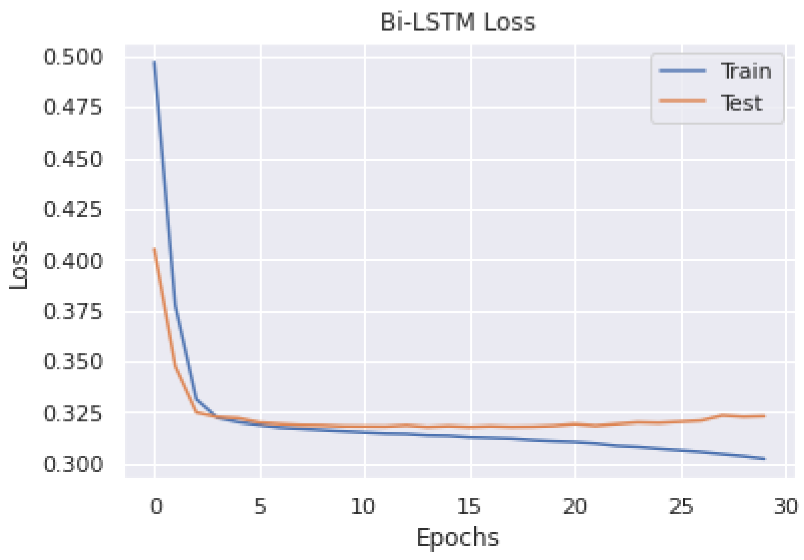
In [126…

```
history = model.fit(train_vec, train_ent_vec,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=[test_vec, test_ent_vec],
          verbose=0, callbacks=[TqdmCallback(verbose=1)])
```

In [127…

```
plot_acc(history)
plot_loss(history)
```





In [128…

```
y_true, y_pred = [], []
test_pred = decode(model.predict(test_vec))
for i in range(len(test_entities)):
    y_true+=test_entities[i]
    y_pred+=test_pred[i][:len(test_entities[i])]
print(classification_report(y_true, y_pred))
```

```
210/210 [==============================] - 6s 26ms/step
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to contro
l this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
              precision    recall  f1-score   support

         LOC       0.25      0.01      0.01      4516
        MISC       0.00      0.00      0.00      2549
           O       0.86      0.99      0.92     95958
         ORG       0.31      0.20      0.24      5287
         PER       0.31      0.07      0.12      6004

    accuracy                           0.84    114314
   macro avg       0.35      0.25      0.26    114314
weighted avg       0.77      0.84      0.79    114314

/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to contro
l this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.8/dist-packages/sklearn/metrics/_classification.py:1318:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to contro
l this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [159…

```
idx = 3
_x_wrd, _x_lab, _x_vec = synth_test_sent[idx], test_entities[idx], test_vec[idx]
_y_pred = decode(model.predict(np.array([_x_vec,])))[0][:len(_x_wrd)]
```

```
org = [(_x_wrd[i], _x_lab[i]) for i in range(len(_x_lab))]
item = [(_x_wrd[i], _y_pred[i]) for i in range(len(_x_wrd))]
```

```
1/1 [==============================] - 0s 71ms/step
```

In [160…

```
doc = dispacy_doc(org, "Generated NER")
html = displacy.render(doc, style="ent", manual=True, jupyter=True)
```

Generated NER

finally **O** the **O** Two **O** of **O** , **O** (**O** Friday **O** AND **O** is **O** who **O** the **O** Jerry **PER** Bevan **PER** David **PER** 0.38 **O** Hungary **LOC** of **O** the **O** surprise **O**

In [161…

```
doc = dispacy_doc(item, "Predicted NER")
html = displacy.render(doc, style="ent", manual=True, jupyter=True)
```

Predicted NER

finally **O** the **O** Two **O** of **O** , **O** (**O** Friday **O** AND **O** is **O** who **O** the **O** Jerry **O** Bevan **ORG** David **ORG** 0.38 **ORG** Hungary **ORG** of **O** the **O** surprise **O**